e-ISSN 2597-4963 dan p-ISSN 1858-4853

SIMULASI ALGORITMA A* DAN DIJKSTRA PADA WAN

Fachrian Anugrah[1)], Nania Nuzulita[2)], Almira Syawli[3)]

Sistem Informasi, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Keputih Sukolilo, Surabaya, 60111
E-Mail : : fachrian15@mhs.is.its.ac.id [1)], nania15@mhs.is.its.ac.id [2)], syawli15@mhs.is.its.ac.id [3)]

ABSTRAK

Perubahan kompleksitas jalur pada internet sejalan dengan meningkatnya perkembangan internet. Sehingga *Network flow optimization* menjadi permasalahan utama terkait dengan pemilihan rute terpendek (*routing protocol*) di balik kompleksitas jalur pada WAN. Algoritma yang digunakan pada *routing protocol* dirancang untuk memiliki kemampuan pencarian rute terpendek. Dengan perkembangan berbagai algoritma saat ini, apakah algoritma penentuan rute terpendek pada graf seperti A* dapat diterapkan untuk penentuan rute terpendek pada jaringan. Fokus pada penelitian ini yaitu mengetahui dan membandingkan proses komputasi dalam pencarian rute terpendek dengan algoritma Dijkstra dan algoritma A* guna mengurangi beban kerja jaringan. Penelitian dilakukan dengan simulasi pada topologi mesh yang menggunakan *fiber link*. Berdasarkan hasil simulasi dapat diketahui bahwa algoritma A* memiliki proses komputasi yang lebih sedikit dan jarak yang dihasilkan lebih pendek dibandingkan dengan algoritma Dijkstra. Oleh karena itu, algoritma A* dapat dipertimbangkan sebagai algoritma pada *routing protocol*.

Kata Kunci : *Wide Area Network* (WAN), *Routing Protocol*, Algoritma Dijkstra, Algoritma A*, Simulasi Proses Komputasi.

1. PENDAHULUAN

Asosiasi Penyelenggara Jasa Internet Indonesisa (APJII) mengungkapkan bahwa jumlah pengguna internet di Indonesia yang tercatat hingga tahun 2016 mencapai 132,7 juta orang. Jumlah tersebut mengalami peningkatan 17% dari tahun 2014 [1]. Peningkatan tersebut sejalan meningkatnya kebutuhan akses internet dari berbagai daerah guna mempermudah komunikasi, penerapan *e-government*, dan sebagainya.

Untuk menyediakan layanan internet antar wilayah geografis yang terpisah, maka digunakan sebuah teknologi *Wide Area Network* (WAN) yang sudah difasilitas oleh banyak penyedia jasa telekomunikasi [1–3]. Salah satu jenis WAN, yaitu jaringan WAN pribadi merupakan solusi bagi suatu perusahaan maupun pemerintahan untuk menghubungkan satu cabang dengan cabang lainnya. Jaringan WAN pribadi dapat mengurangi ketidak pastian atau ketidak andalan bandwidth jaringan internet sekarang ini [4].

Adapun infrastruktur jaringan yang dibutuhkan untuk menghubungkan antar segmen atau wilayah dalam jaringan WAN meliputi *router*, *bridge*, *fiber optic*, dan sebagainya. Peningkatan jumlah infrastruktur jaringan akan berdampak pada kompleksitas jalur komunikasi di dalam jaringan.

Kompleksitas jalur komunikasi mencakup kebutuhan *bandwidth*, konfigurasi fisik jaringan, waktu pemrosesan paket dari setiap *device*, penentuan rute pengiriman paket, dan sebagainya. *Network flow optimization* adalah salah satu permasalahan utama terkait dengan pemilihan rute terpendek [5]. Konsep dasar pemilihan rute terpendek pada jaringan adalah bagaimana memodelkan rute terpendek dengan biaya perjalanan rendah yang kemudian dikenal sebagai *routing protocol*.

Saat ini telah berkembang *routing protocol* seperti RIP (*Routing Information Protoco*l) [6], OSPF (*Open Shortest Path First*) [7], dan *green routing protocols* [8]. Dari setiap *routing protocol* tersebut digunakan sebuah algoritma untuk penentuan rute, salah satunya adalah algoritma Dijkstra yang diterapkan pada SPF (*Shortest Path First*) [9,10].

Algoritma Dijkstra merupakan algoritma umum yang digunakan dalam penentuan rute terpendek pada graf [5]. Untuk lebih mengoptimalkan algoritma Dijkstra pada kasus jaringan, sudah dilakukan penelitian lain dengan memodifikasi algoritma Dijkstra tersebut seperti *green routing protocols* [8,11] atau algoritma genetik dan *Particle Swarm Optimization* (PSO) [12].

Router dalam jaringan WAN dapat diinterpretasikan sebagai graf sehingga memungkinkan penggunaan algoritma penentuan rute terpendek seperti algoritma A* [13–16] dan *Floyd-Warshall* [14]. Oleh karena itu yang menjadi fokus pada penelitian ini adalah mensimulasikan perbandingan algoritma A* sebagai salah satu algoritma graf yang belum diterapkan di *routing protocol* dimana algoritma tersebut memiliki kemampuan komputasi lebih kecil dari Dijkstra [13,14] dan membandingkannya dengan algoritma Dijkstra yang digunakan sebagai standar *routing protocol* saat ini [9,10]. Dari hasil penelitian ini

e-ISSN 2597-4963 dan p-ISSN 1858-4853

diharapkan mampu berkontribusi dalam *network flow optimization* terutama dalam proses komputasi pencarian rute dengan jarak terpendek yang dibutuhkan mencapai tujuan, sehingga mampu mengurangi beban kerja jaringan.

Bagian lain dari dokumen ini disusun sebagai berikut: penjelasan detail terkait penelitian sebelumnya, dasar teori algoritma dijkstra dan algoritma A* serta simulasi algoritma Dijkstra dan algoritma A* dijelaskan pada bagian 2, hasil penelitian beserta pembahasan dijelaskan pada bagian 3, serta kesimpulan dan saran dari penelitian ini dijelaskan pada bagian 4.

2. TINJAUAN PUSTAKA

A. Algoritma Dijkstra

Algoritma Dijkstra ditemukan pada tahun 1956 oleh seorang ilmuan komputer dari Belanda bernama Edsger Wybe Dijkstra yang kemudian dipublikasikan pada tahun 1959. Algoritma ini merupakan algoritma untuk menemukan jalur terpendek antar graph. Secara sederhana, algoritma ini digambarkan dengan sebuah simpul yang berperan sebagai simpul "sumber" dan menemukan jalan tercepat dari sumber ke semua node di dalam graph yang kemudian menjadi pohon jalur tecepat (*shortest-path tree*) [17].

Algoritma Dijkstra menggunakan strategi dari algoritma *greedy* yaitu memilih yang terbaik dari semua pilihan yang ada. Algoritma ini mengambil sisi dengan bobot lebih kecil dari simpul belum terpilih yang terhubung dengan satu simpul yang sudah terpilih. Lintasan yang terbentuk dari simpul asal ke simpul asal ke simpul baru adalah lintasan yang terpendek. Gambar 1 berikut menunjukkan pseudocode dari algoritma Dijkstra [9].

```
function Dijkstra (input M :
matriks, a : simpul awal) -> tabel
{
  Mencari lintasan terpendek dari
impul awal a ke semua simpul
alinnya
Masukan: matriks ketetanggaan (M)
dari graf berbobot G dan simpul
awal a
Keluaran: tabel D yang berisi
panjang lintasan terpendek dari a
ke semua simpul
lainnya
}
Deklarasi
  D, S: tabel
  i: integer
Algoritma
 {Langkah 0 inisialisasi}
 For I <- 1 to n do
   S[i] <- 0
   D [i] <- m[a,i]
 Endfor
{Langkah 1}
 S[a] <- 1 {karena simpul a adalah
simpul asal lintasan terpendek,
jadi simpul a sudah pasti terpilih
dalam lintasan terpendek}
 D[a] <- tak hingga {tidak ada
lintasan terpendek dari simpul a
ke a}
{langkah 2, 3, …, n-1}
For I <- 2 to n-1 do
Cari j sedemikian hingga sehingga
S[j] = 0 dan D[j] = Minimum {D[1],
D[2], …, D[n]}
S[j] <- 1 {simpul j sudah terpilih
ke dalam lintasan terpendek}
Hitung D[i] yang baru dari a k
simpul i yang bulan elemen S
dengan cara:
D[i]<-Minimum{D[i],(D[j]+M[j,i]}
 Endfor
 Return D
```

Gambar 1. Pseudecode Algoritma Dijkstra

B. Algoritma A*

Algoritma A* (A Star) pertama kali diperkenalkan pada tahun 1968 oleh Peter Hart, Nils Nilsson, dan Bertran Raphael. Algoritma ini merupakan algoritma pencarian *graph* yang digunakan untuk menemukan jalur dengan biaya pengeluaran paling kecil dari titik awal hingga titik tujuan yang diharapkan. Algoritma ini merupakan *best first search* dengan estimasi solusi biaya terkecil dengan jarak dan nilai heuristik (kriteria untuk menentukan alternatif) sebagai dasar pertimbangan [16].

Rumus yang digunakan oleh Algoritma A Star direpresentasikan pada persamaan (1) sebagai berikut [13]:

$$f(n) = g(n) + h(n) \quad (1)$$

Dimana:

f(n) : perkiraan jalur terpendek sementara dari *node* n.
g(n) : total jarak yang didapat dari *node* awal ke *node* tujuan
h(n) : perkiraan biaya dari *node* n ke *node* tujuan

Algoritma A* menggunakan koordinat yang digunakan pada persamaan *Ecluidian Distance* untuk memperoleh perkiraan biaya dari *node* n ke *node* tujuan. Biaya tersebut merupakan *h(n)* yang tercantum pada persamaan sebelumnya. Persamaan *Ecluidian Distance* yang dituangkan pada persamaan (2).

$$h(n) = \sqrt{(tujuan.x - n.x)^2 + (tujuan.y - n.y)^2} \quad (2)$$

Dimana:

h(n) : perkiraan jarak dari *node* n ke tujuan.
n.x : koordinat x dari *node* n
n.y : koordinat y dari *node* n
tujuan.x : koordinat x dari *node* tujuan
tujuan.y : koordinat y dari *node* tujuan

Gambar 2 berikut menunjukkan *pseudocode* dari algoritma A* [18] .

```
Fungsi A*(awal, tujuan)
Var himp tertutup <- himp_kosong
Var q <- buat_antrian(awal)
```

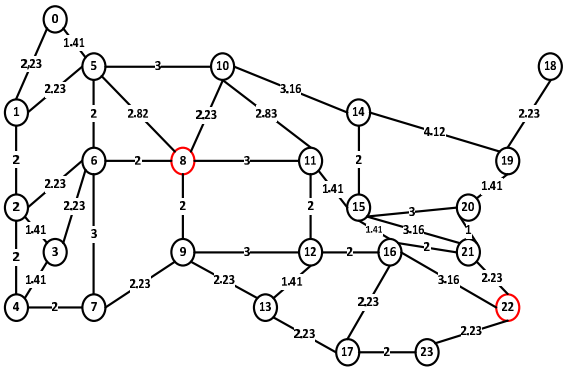
e-ISSN 2597-4963 dan p-ISSN 1858-4853

```
Selama (q tidak kosong)
{
  X = node ada di dalam q dengan nilai f terkecil
    If (x = tujuan)
    {
      Ditemukan rute terpendek
    }
  Hapus x dari q
  Tambahkan x ke himp_tertutup
  Untuk setiap y adalah node tetangga (x)
  {
    Jika y ada dalam himp_tertutup atau adalah unwalkable
    {
      Continue
    }
    Jika y belum ada pada q
    {
      Tambahkan y pada q dan set nilai F, G, H
    }
      Jika y terdapat pada q
      {
        Periksa nilai G
        Jika (ada y yang memiliki nilai g terkecil)
        {
          Ganti x dengan y, update nilai F, G, H
        }
      }
  }
}
```

Gambar 2. Pseudecode Algoritma A*

C. Simulasi Algoritma Dijkstra dan A*

Untuk mengetahui perbandingan komputasi dengan kedua algortima pada jaringan WAN, digunakanlah topologi mesh yang ditunjukkan pada Gambar 3 yang terdiri dari 24 *node* dan 42 *fiber link*. Angka di setiap garis yang menghubungkan antar *node* menunjukkan jarak antar *node* dengan rasio 1:50 km. sementara setiap *node* adalah *router* yang berada dalam jaringan WAN Dalam penelitian ini, diasumsikan bahwa *node* awal berada pada *node* 8 dan *node* tujuan akhir adalah *node* 22.



Gambar 3. Rancangan Topologi.

1. Simulasi dengan Algoritma Dijkstra

Pada dasarnya, algoritma Djikstra memiliki tujuan untuk menemukan jalur terdekat dari *node* awal menuju *node* tujuan. Jalur terdekat didapatkan dengan cara mencari pada setiap *node* yang saling terhubung kemudian menentukan jarak yang paling pendek sebagai pilihan terbaik. Kemudian jarak terpendek tersebut akan dijumlahkan dengan *node* ke-n hingga mencapai *node* tujuan.

Pada penelitian ini seperti yang disajikan pada Gambar 3, rute yang dicari adalah rute dari *node* 8 menuju *node* 22. Algoritma Djikstra memulai pencarian dari *node* 8 menuju ke *node* 5 dengan jarak sebesar 2.82, *node* 6 dengan jarak sebesar 2, *node* 9 dengan jarak sebesar 2, dan *node* 10 dengan jarak sebesar 3. Untuk menentukan *node* yang akan dilalui selanjutnya maka dilakukan dengan cara mencari jarak yang terdekat dari *node* yang belum pernah dilewati yaitu *node* 6. Walaupun hasil jarak dari *node* 6 dan *node* 9 sama, yang akan dijadikan sebagai jarak terkecil adalah *node* 6 karena *node* 6 diproses terlebih dahulu daripada *node* 9.

Setelah *node* yang akan dilalui selanjutnya ditemukan, kemudian dilakukan perhitungan serupa dengan menjumlahkan total jarak yang didapat dari *node* sebelumnya (*node* 6 sebesar 2) dengan *node* selanjutnya yaitu *node* 2 sebesar 4.24, *node* 3 sebesar 4.23, *node* 7 sebesar 5, dan *node* 5 sebesar 4. Namun karena terdapat jarak yang lebih dekat menuju *node* 5 dari *node* 8 sebesar 2.82 daripada dari *node* 6 sebesar 4, sehingga didapatkan jarak terdekat untuk menuju *node* 5 melalui *node* 8. Tahap perhitungan dengan perkiraan jalur terpendek terus diulang hingga menemukan *node* tujuannya adalah *node* 22.

2. Simulasi dengan Algoritma A*

Proses komputasi yang dilakukan dengan algoritma A* dijelaskan pada sub bab ini. Dapat dilihat dalam gambar bahwa *node* 8 terhubung dengan *node* 5, 6, 9, 10, dan 11. Dengan menggunakan persamaan *Ecluidian Distance*, dihitung perkiraan biaya dari *node* 8 ke *node* 5, 6, 9, 10, dan 11. Sebagai contoh, *node* 8 ke *node* 5 sebagai berikut:

$$h(5) = \sqrt{((13-3)^2 + (2-7)^2)} = 11.18$$

Setelah diperoleh *ecluidian distance* dari *node* 8 ke *node* 5, hasilnya ditambahkan dengan jarak sebenarnya untuk mendapatkan perkiraan jalur terpendek sebagai berikut:

$$f(5) = 2.82 + 11.18 = 14$$

Setelah nilai f(5), f(6), f(9), f(10), dan f(11) didapat, kemudian dipilih nilai f(n) yang terkecil dari nilai f(n) yang belum pernah dipilih yaitu f(5) = 14, f(6) = 12.44, f(9) = 10.06, f(10) = 10.83, dan f(11) = 8.83 sehingga nilai f(n) terkecil yang digunakan sebagai *node* selanjutnya adalah *node* 11.

Kemudian ditetapkan *node* 11 sebagai *node* awal dan *node* 15 dan 12 sebagai *node* tujuan selanjutnya. Tahap perhitungan dengan persamaan *ecluidian distance* dan perkiraan jalur terpendek terus diulang hingga *node* tujuannya adalah tujuan dari kasus ini yaitu 22.

3. HASIL DAN PEMBAHASAN

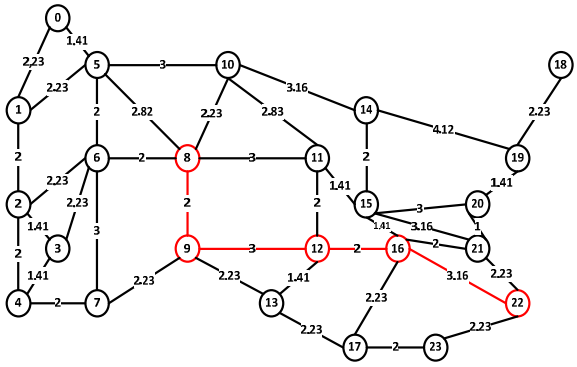
Berdasarkan simulasi yang telah dijelaskan pada bagian 2.4, maka diperoleh hasil komputasi pencarian rute dari setiap algoritma untuk studi kasus penelitian ini yang ditunjukkan pada Tabel 1 untuk komputasi algoritma dijkstra dan Tabel 2 untuk komputasi algoritma A*.

Tabel 1. Proses Komputasi dengan Algoritma Dijkstra

Proses Komputasi	*Node*		Hasil	Urutan
	Asal	Tujuan		
1	8	5	2,82	3
		6	2	1
		9	2	2
		10	3	4
2	6	2	4,23	6
		3	4,23	7
		5	4	
		7	5	
3	9	7	4,23	8
		12	5	10
		13	4,23	9
4	5	0	4,23	5
		1	5,05	11
		10	5,82	
5	10	11	5,83	13
		14	6,16	14
6	0	1	6,46	
7	2	1	6,23	
		3	5,64	
		4	6,23	
8	3	4	5,64	12
9	7	4	6,23	
10	13	12	5,64	
		17	6,46	15
11	12	11	7	
		16	7	16
12	1			
13	4			
14	11	15	7,24	
15	14	15	8,16	
		19	10,28	
16	17	16	8,69	
		23	8,46	
17	16	21	9	
		22	10,16	17

Berdasarkan perhitungan Algoritma Djikstra, didapatkan total jarak dari *node* 8 menuju *node* 22 sebesar 10.16 dengan total proses komputasi sebanyak 17 proses untuk menemukan *node* tujuannya seperti yang dijelaskan pada Tabel 1. *Node-node* yang dilewati dari *node* 8 agar mencapai 22 diantaranya adalah *node* 8 - 9 - 12 - 16 - 22. Jalur ini didapatkan dari proses 1 *node* 8 menuju 9, proses 3 *node* 9 menuju 12, proses 11 *node* 12 menuju 16, dan proses 17 *node* 16 menuju *node* 22.

Node-node tersebut diambil dengan memperhatikan keseluruhan kemungkinan jarak terkecil yang belum pernah dilalui sebelumnya. Pada proses pertama, didapatkan bahwa *node* 6 merupakan *node* dengan jarak terkecil dari 8. Berarti *node* yang diproses selanjutnya adalah *node* 6. Kemudian perhitungan dilanjutkan dari *node* 6 dengan yang terhubung dengannya yaitu *node* 2, 3, 5, dan 7 dan ditambahkan jaraknya dari *node* 8 ke *node* 6. Setelah diperoleh hasilnya, semua hasil tersebut dibandingkan dengan hasil dari *node* yang belum terlewati, termasuk dengan hasil terkecil dari proses yang pertama. Karena jarak yang terkecil selanjutnya dimiliki oleh *node* 8 ke *node* 9, maka pencarian jalur terkecil dimulai dari *node* 9. Hal ini terus dilakukan secara berulang hingga algoritma Dijkstra mencapai tujuannya. Jalur yang didapat dari hasil komputasi dengan algoritma Dijkstra digambarkan pada Gambar 4.



Gambar 4. Jalur Terpendek dengan Algoritma Dijkstra.

Hasil dari algoritma Dijkstra bisa saja menghasilkan jalur yang bukan merupakan jarak total yang terdekat karena jarak yang dipertimbangkan bukan merupakan total jarak keseluruhan dengan nilai terkecil. Namun algoritma Dijkstra memproses apa yang menurutnya paling kecil dari *node* yang sedang diprosesnya.

Tabel 1 juga menunjukkan bahwa algoritma Dijkstra membutuhkan perulangan yang banyak untuk memeriksa semua jarak dari satu *node* ke *node* yang lainnya. Walau secara otomatis beban komputasi lebih tinggi, algoritma Dijkstra merangkul lebih banyak *node* untuk 'dikenali' oleh *node* tetangganya. Dalam konsep *routing*, dapat diartikan satu *router* dapat mengetahui *router* apa saja yang terhubung dengannya sehingga algoritma Dijkstra cocok digunakan untuk topologi jaringan yang hanya membutuhkan sampainya data tanpa

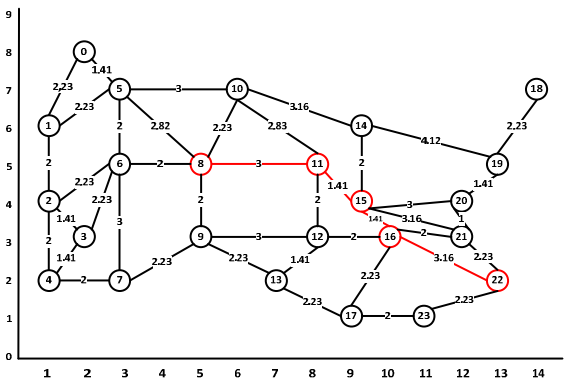
memperhatikan ke-optimalan jarak yang ditempuhnya.

Tabel 2. Proses Komputasi dengan Algoritma A*

Proses Komputasi	*Node* Asal	*Node* Tujuan	Hasil	Urutan
1	8	5	14,00	1
		6	12,44	
		9	10,06	
		10	10,83	
		11	8,83	
2	11	10	14,43	2
		12	10,10	
		15	8,88	
3	15	14	12,07	3
		16	8,98	
		20	9,65	
		21	8,98	
4	16	12	12,92	4
		17	12,17	
		21	9,23	
		22	8,98	

Berbeda dengan Algoritma Djikstra, pengambilan keputusan untuk menentukan *node* berikutnya yang akan dilalui pada Algoritma A* tidak berdasarkan jarak saja g(n), namun juga berdasarkan jarak estimasi dari *node* berikutnya menuju *node* tujuan h(n) dengan menghitung Persamaan *Ecluidian Distance*. Jarak estimasi ini didapatkan dengan menarik garis lurus dari *node* n menuju *node* tujuan kemudian menjumlahkannya dengan total jarak yang didapat dari *node* awal menuju *node* n.

Berdasarkan perhitungan Algoritma A*, didapatkan total jarak dari *node* 8 menuju *node* 22 sebesar 8.98 dengan total proses komputasi sebanyak 4 proses untuk menemukan *node* tujuannya seperti yang dijelaskan pada Tabel 2. *Node-node* yang dilewati diantaranya adalah *node* 8 - 11 - 15 - 16 - 22. *Node-node* ini didapatkan dari proses 1 *node* 8 menuju 11, proses 2 *node* 11 menuju 15, proses 3 *node* 15 menuju 16, dan proses 4 *node* 16 menuju *node* 22. Jalur yang didapat dari hasil komputasi dengan algoritma A* digambarkan pada Gambar 5.



Gambar 5. Jalur Terpendek dengan Algoritma A*

Hasil dari algoritma A* merupakan jalur dengan total jarak yang paling kecil. Hal ini dikarenakan algoritma A* memperhitungkan dekat atau tidaknya posisi *node* satu dengan *node* lainnya bukan hanya melalui jalur yang tersedia diantaranya, melainkan dari posisi sesungguhnya antar *node* tersebut. Dari penjelasan berikut, dapat dikatakan bahwa algoritma A* cocok untuk topologi jaringan yang berfokus untuk mengirimkan satu paket tertentu atau dapat dikatakan yang berfokus pada satu tujuan saja. Sehingga untuk paket dengan tujuan yang berbeda pada penerapan algoritma A*, *router* harus melakukan *routing* ulang.

4. KESIMPULAN DAN SARAN

A. Kesimpulan

Fokus penelitian ini yaitu mensimulasikan dua shortest path algorithm yaitu Dijkstra dan A* untuk membandingkan hasil komputasi dari keduanya. Berdasarkan hasil simulasi didapatkan bahwa algoritma A* memiliki komputasi yang lebih cepat dibandingkan dengan Dijkstra. Hal ini ditunjukkan dengan jumlah proses pemilihan *node* yang akan dilewati lebih sedikit dibandingkan dengan Dijkstra. Hal tersebut dapat berdampak pada waktu yang dibutuhkan untuk memperbaharui *routing table*, sehingga memungkinan waktu yang dibutuhkan lebih cepat.

Walaupun algoritma A* membutuhkan waktu yang relatif lebih singkat daripada algoritma Dijkstra, algoritma A* sangat berfokus pada *node* yang menjadi tujuannya. Sehingga algoritma A* menjadi sangat baik ketika setiap kali ada paket yang mau dikirim, *router* memperbaharui isi *routing table*-nya. Sementara pada algoritma Dijkstra, *routing table* terisi lebih banyak informasi karena perulangan dilakukan berkali-kali sehingga satu *router* dapat mengenali *router* lain yang terhubung padanya.

B. Saran

Penelitian selanjutnya dapat dilakukan studi mengenai waktu yang dibutuhkan sebuah *router* dalam jaringan untuk memperoleh *table routing* dengan kedua algoritma ini. Kedua algoritma ini dapat juga diterapkan pada beberapa percobaan dalam topologi yang sama namun dengan *node* awal dan tujuan yang berbeda dan menghitung waktu yang dibutuhkan *router* untuk melakukan routing hingga waktu sampainya paket ke tujuan sehingga dapat diperoleh algoritma mana yang lebih efektif maupun efisien. Penelitian lain yang juga dapat dilakukan adalah membandingkan algoritma lainnya seperti *sleep-scheduling* dan penggunaan medium lain seperti wireless dengan bantuan simulator jaringan agar hasil komputasi lebih akurat. Selain itu juga perlu dipertimbangkan *traffic* yang ada pada jaringan.

5. DAFTAR PUSTAKA

[1] APJII, "Asosiasi Penyelenggara Jasa Internet Indonesia." [Online]. Available: https://www.apjii.or.id/. [Accessed: 26-Nov-2016].

[2] Larry L. Learn, "NETWORKS: A Review of Their Technology, Architecture and Implementation," *Libr. Hi Tech*, vol. 6, no. 2, pp. 19–49, Feb. 1988.

[3] H. Kaur and R. K. Gurm, "A SURVEY ON WAN TECHNOLOGIES," *Int. J. Multi Discip. Eng. Bus. Manag. IJMDEBM*, vol. 3, no. 2, pp. 62–65, 2013.

[4] W. Li, D. Guo, K. Li, H. Qi, and J. Zhang, "iDaaS: Inter-Datacenter Network as a Service," *IEEE Trans. Parallel Distrib. Syst.*, vol. PP, no. 99, pp. 1–1, 2015.

[5] L. Qi and M. Schneider, "Trafforithm: A traffic-aware shortest path algorithm in real road networks with traffic influence factors," in *2015 1st International Conference on Geographical Information Systems Theory, Applications and Management (GISTAM)*, 2015, pp. 1–8.

[6] G. S. Malkin, "RIP Version 2." [Online]. Available: https://tools.ietf.org/html/rfc2453.html. [Accessed: 22-Nov-2016].

[7] J. Moy, "OSPF Version 2." [Online]. Available: https://tools.ietf.org/html/rfc2178#page-206. [Accessed: 22-Nov-2016].

[8] F. Dabaghi, Z. Movahedi, and R. Langar, "A survey on green routing protocols using sleep-scheduling in wired networks," *J. Netw. Comput. Appl.*, vol. 77, pp. 106–122, Jan. 2017.

[9] M. G. Mahfudhi, "Penerapan Algoritma Dijkstra pada Link State Routing Protocol untuk Mencari Jalur Terpendek," *Strategi Algoritma*, 2010.

[11] A. Cianfrani, V. Eramo, M. Listanti, M. Marazza, and E. Vittorini, "An Energy Saving Routing Algorithm for a Green OSPF Protocol," in *2010 INFOCOM IEEE Conference on Computer Communications Workshops*, 2010, pp. 1–5.

[12] J. Galán-Jiménez and A. Gazo-Cervero, "Using bio-inspired algorithms for energy levels assessment in energy efficient wired communication networks," *J. Netw. Comput. Appl.*, vol. 37, pp. 171–185, Jan. 2014.

[13] A. Fachrian, "Rancang Bangun Penentuan Rute Terpendek Angkutan Kota Di Surabaya Berbasis Android Dengan Menggunakan Algoritma A*," skripsi, UNIVERSITAS AIRLANGGA, 2015.

[14] M. A. Djojo and Karyono, "Pengukuran Beban Komputasi Algoritma Dijkstra, A , dan Floyd- Warshall pada Perangkat Android," *1 Vol5 Sept. 2013*, vol. 0, no. 0, Sep. 2013.

[15] W. Setiawan, "Pembahasan Pencarian Lintasan Terpendek Menggunakan Algoritma Dijkstra dan A*," *J. Mat. ITB*.

[16] Y. Yuliani and F. Agus, "WEBGIS PENCARIAN RUTE TERPENDEK MENGGUNAKAN ALGORITM A STAR (A*) (Studi Kasus: Kota Bontang) 1) Yuliani," *J. Inform. Mulawarman JIM*, vol. 8, no. 2, pp. 50–55, Jun. 2016.

[17] "Dijkstra's algorithm," *Wikipedia*. 09-Nov-2016.

[18] P. S. N. Ananda, S. Wahjuni, and E. P. Giri, "Penentuan Rute Terpendek Menggunakan Variasi Fungsi Heuristik Algoritme A* Pada Mobile Devices," *J. Ilm. Ilmu Komput.*, vol. 8, no. 2, 2010.